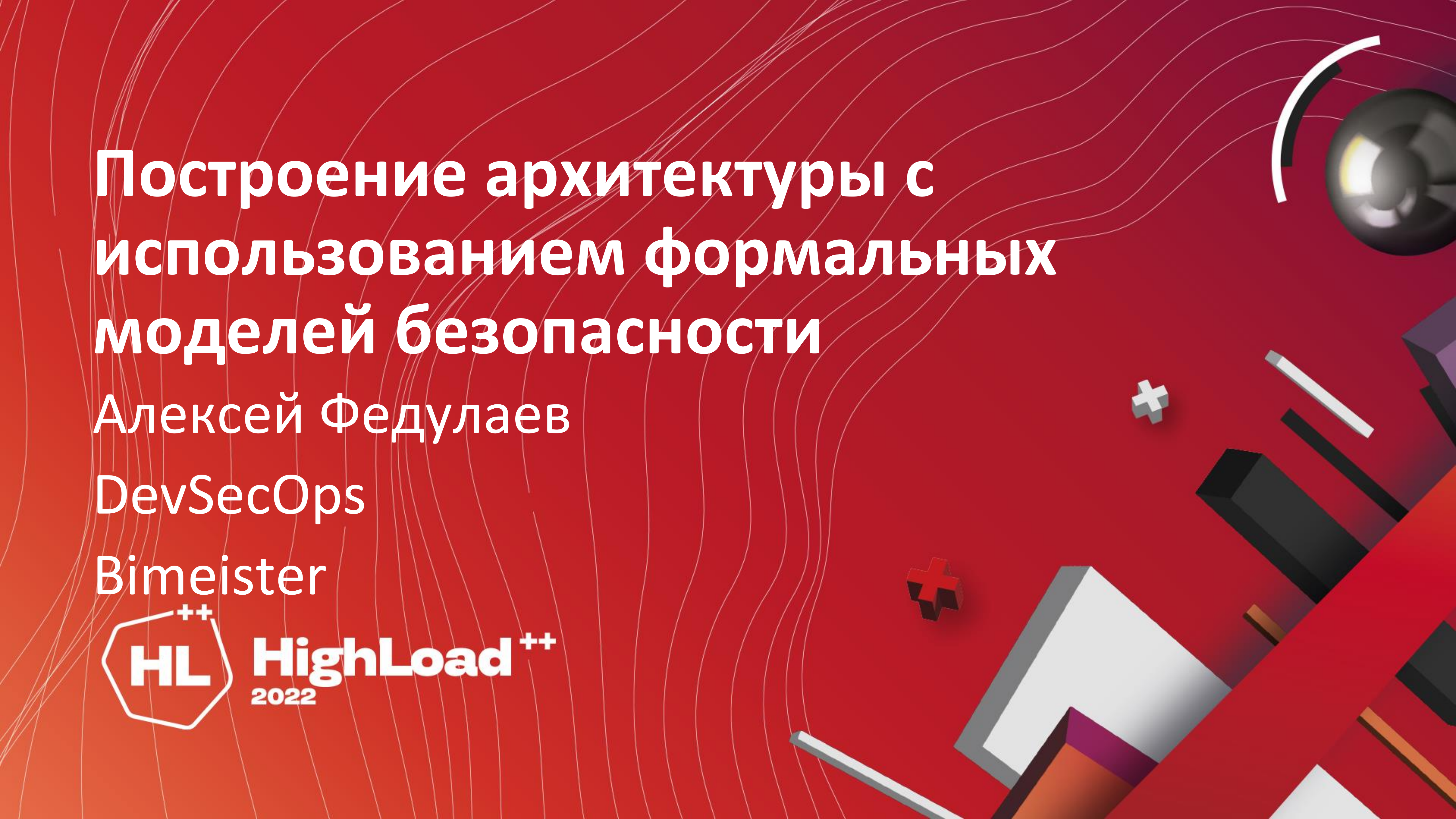


Построение архитектуры с использованием формальных моделей безопасности
Алексей Федулаев
DevSecOps
Bimeister
HL
HighLoad++
2022

Whoami

11

лет в информационной безопасности

DevSecOps

в продуктовой IT-компании

Bimeister –

разработка, инжиниринг и внедрение инновационных цифровых решений для промышленности





О чем сегодня поговорим?

- Как максимально сдвинуться в ShiftLeft?
- Почему важно поддерживать архитектуру в актуальном состоянии?
- Как это можно делать?
- Как выявлять нарушения безопасности в архитектуре?



ShiftLeft

Sec
CODE
PLAN
Dev
BUILD
TEST
RELEASE
DEPLOY
Ops
OPERATE
MONITOR



Где у вас документация?

Тимлид:

@ithumor

— Я и есть документация.



Архитектура продукта

Это компоненты, из которых продукт состоит, и связь между ними

- Позволяет выявлять и исправлять ошибки на этапе идеи
- Однозначная интерпретация идеи
- Эталонная модель для верификации
- Легкий онбординг для новых людей в команде



Классический пример

Cameras

Users

Cam API

Streamer

User API

Account

Notification

Reciever

User mgr

Storage mgr

Payments

Cam mgr

Transactions

Stream mngr

Bank API

Bank



Data Flow Diagram

User

Order information

Orders

Fullfilment

Orders

Product

Shipping information

Process order

Username, adress

Username, adress

Ship products

Billing

Users

Product

Username, adress

Invoices

Invoices, statements/ payments, inquiries

Collect payment

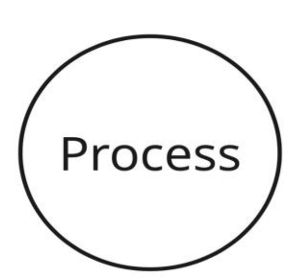
Users

Billing



Обозначения

- Процесс
- Хранилища данных
- Внешняя по отношению к описываемой системе сущность
- Поток данных



Data Store

External Entity

Data Flow



Правила построения

- Внешние сущности располагаются по краям
- Хранилища данных не могут передавать данные между собой без процесса
- Каждый процесс и хранилища данных должны иметь входные и выходные данные



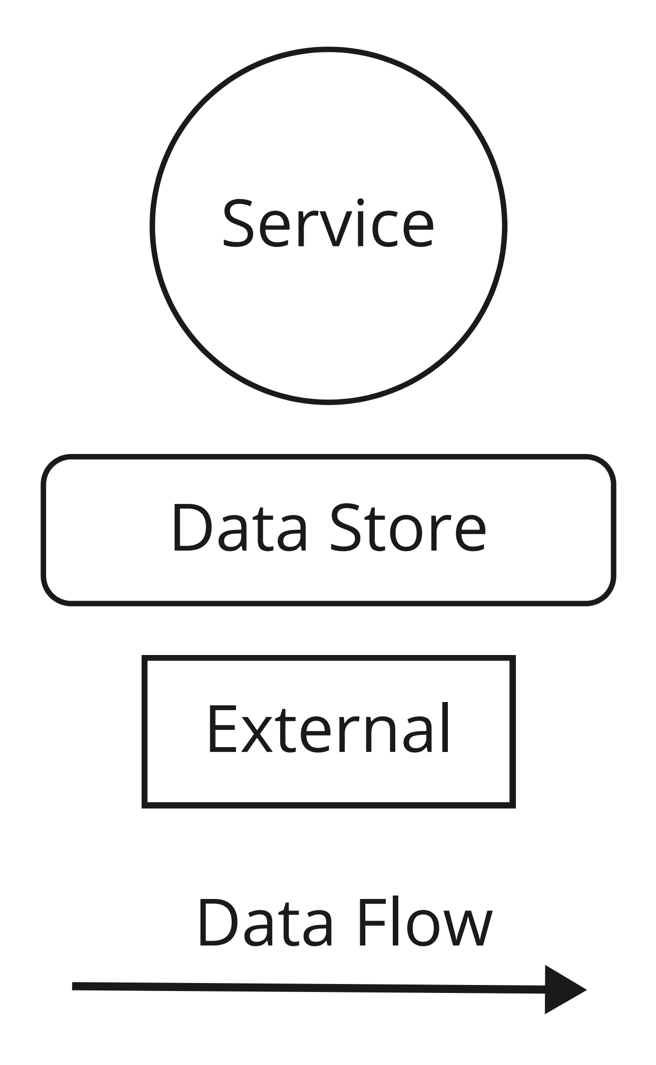
Почему DFD?

- Простота
- Удобство масштабирования
- Основной упор на потоки данных



Как это применить к сервисам?

- В качестве процесса будет выступать микросервис
- В качестве хранилищ любые сущности для хранения данных (БД, файлы и др.)
- В качестве внешних сущностей – пользователи, системы





Тестовая архитектура

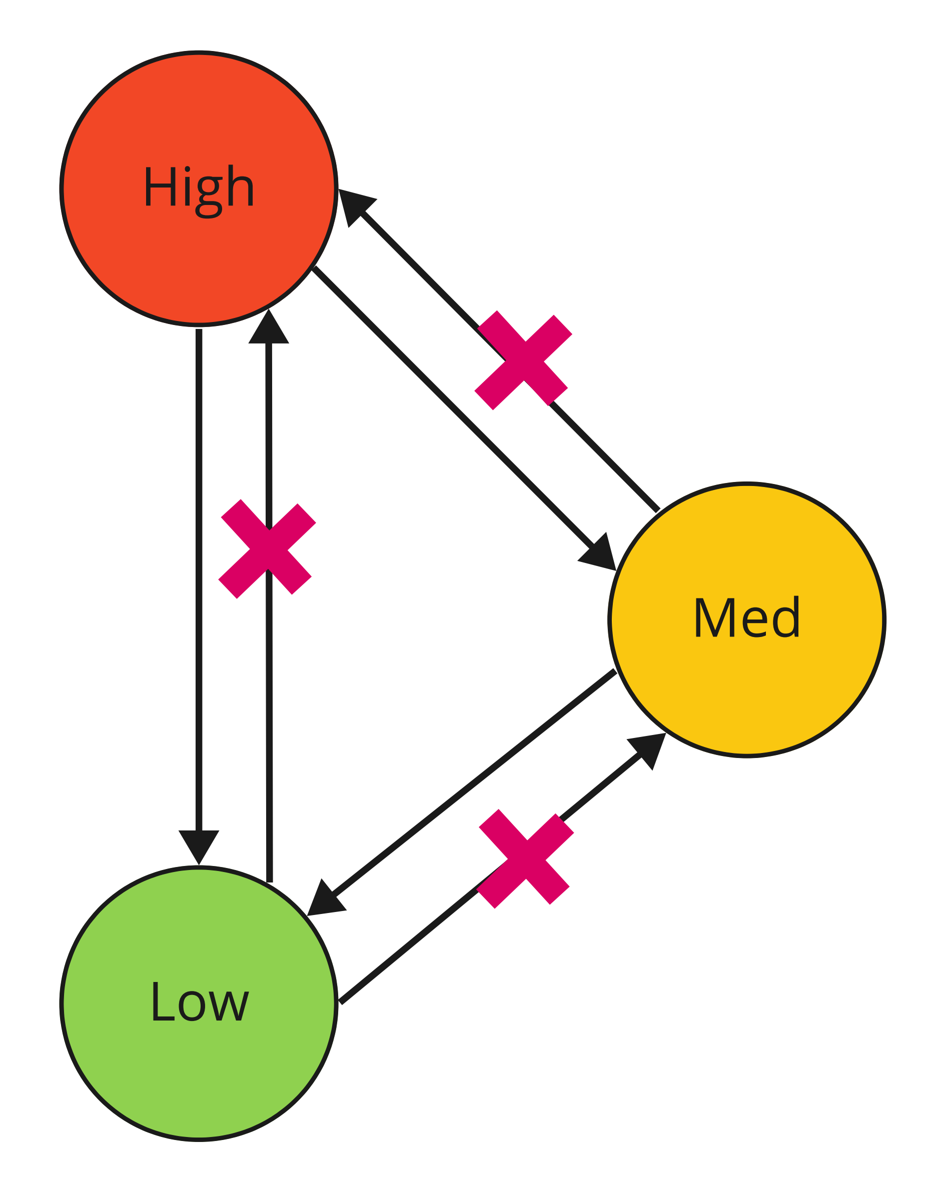
Cameras
Users
Notification
Reciever
Cam API
User API
Account
Transactions
Bank API
Streamer
Bank
Storage mgr
Cam mgr
Stream mngr
User mgr
Payments



Модель Биба

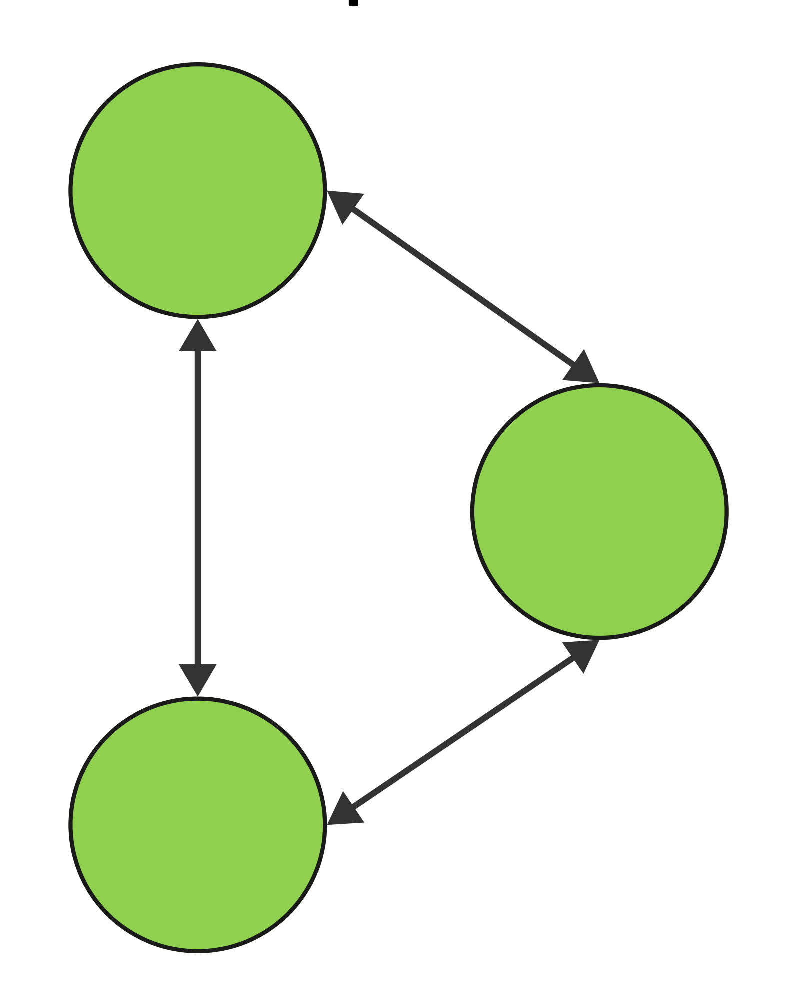
Вводит разные уровни целостности

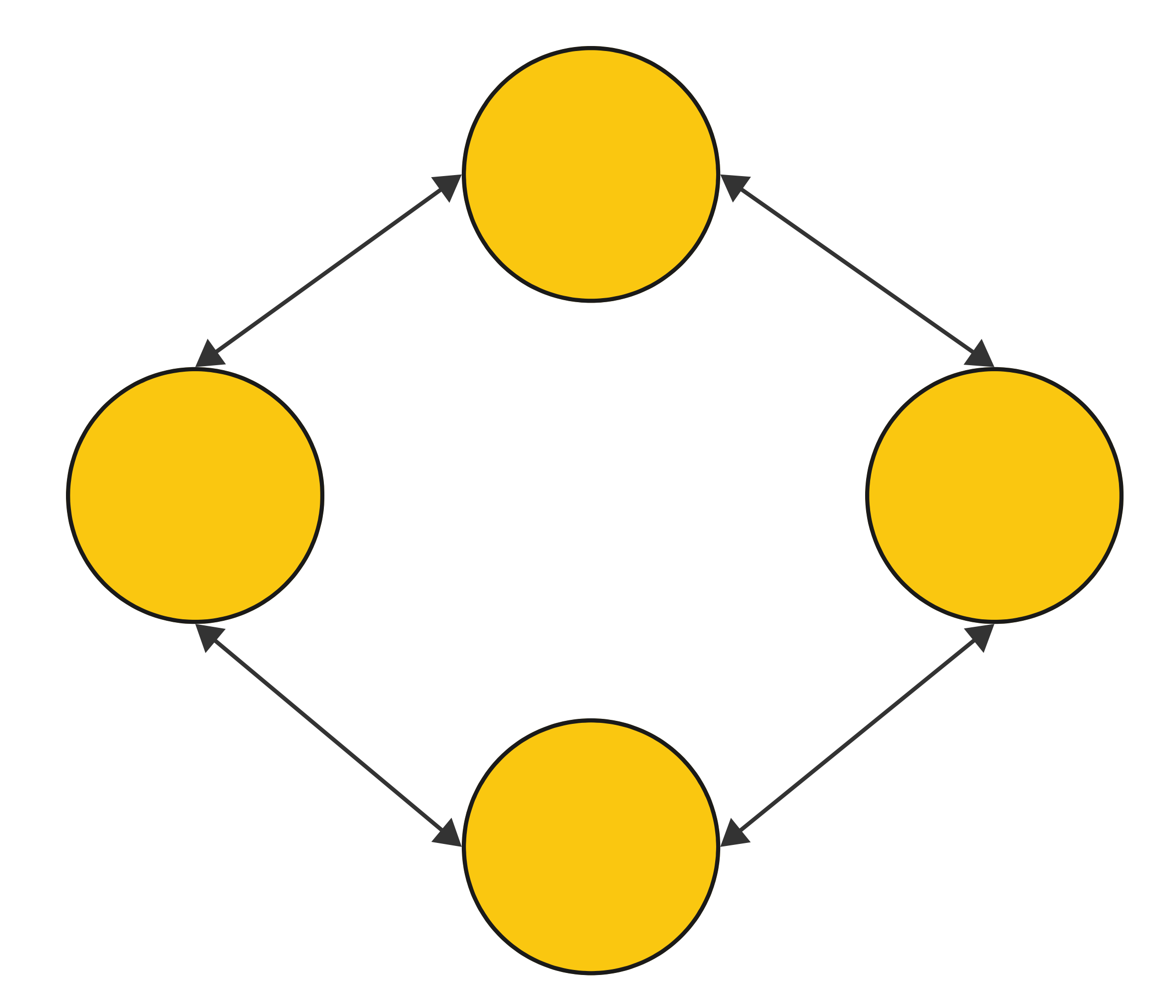
- Нельзя читать вверх
- Можно писать вниз





Изолированная система







Система в вакууме



Как повысить доверие к данным?

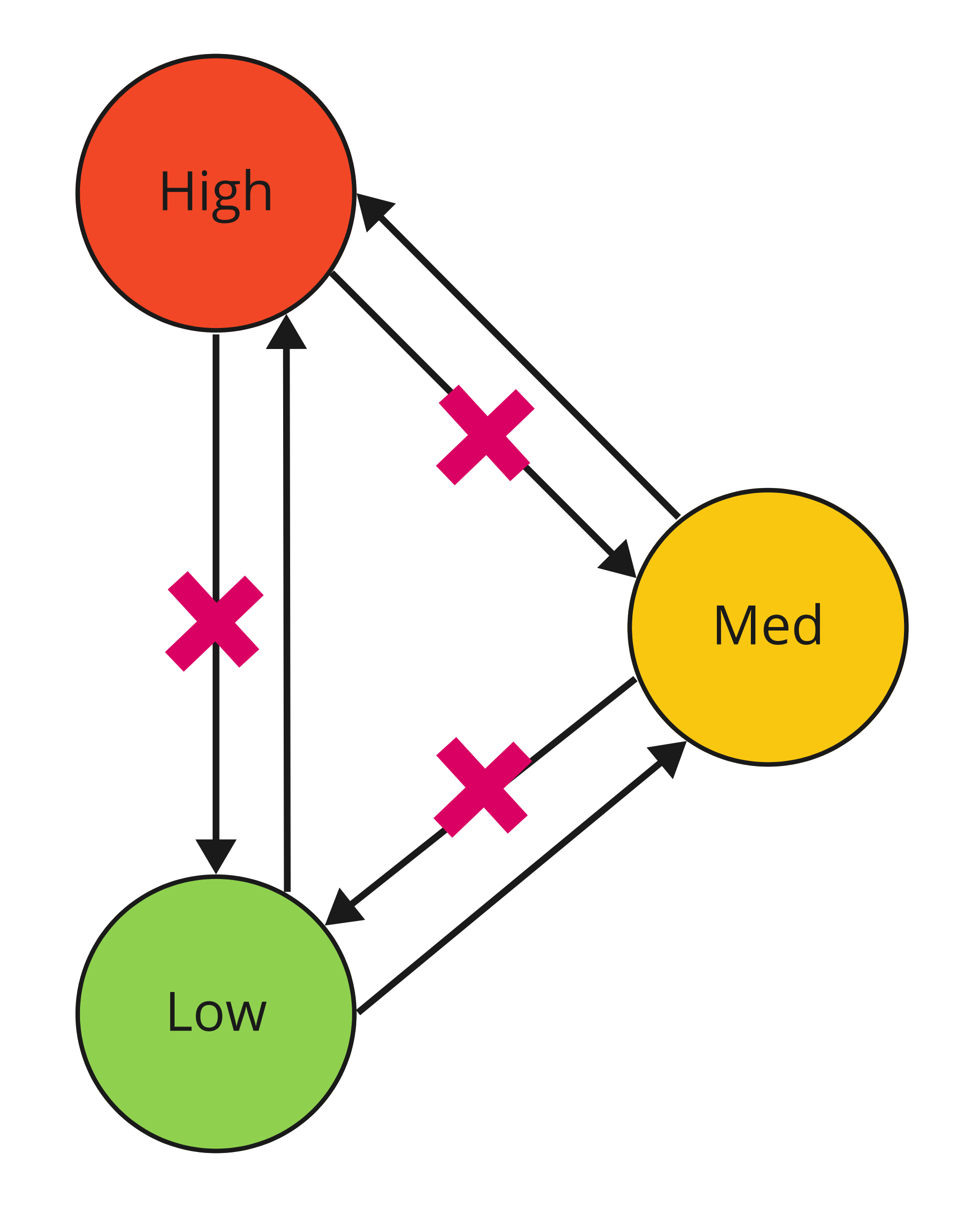
- Типизация
- Санитизация
- Валидация



Модель Белла ЛаПадула

Вводит разные уровни конфиденциальности

- Нельзя писать вниз
- Можно читать вверх





Понижаем уровень доступа к данным

- Типизация
- Санитизация
- Валидация

Service 2 ← ? ← Service 1

Service 2 ← ☆ ← Service 1



Соглашение об обозначениях

- Черным цветом – минимальный уровень
- Зеленым цветом – низкий уровень
- Желтым цветом – средний уровень
- Красным цветом – высокий уровень
- Фиолетовым – нарушения целостности
- Голубым – нарушения конфиденциальности

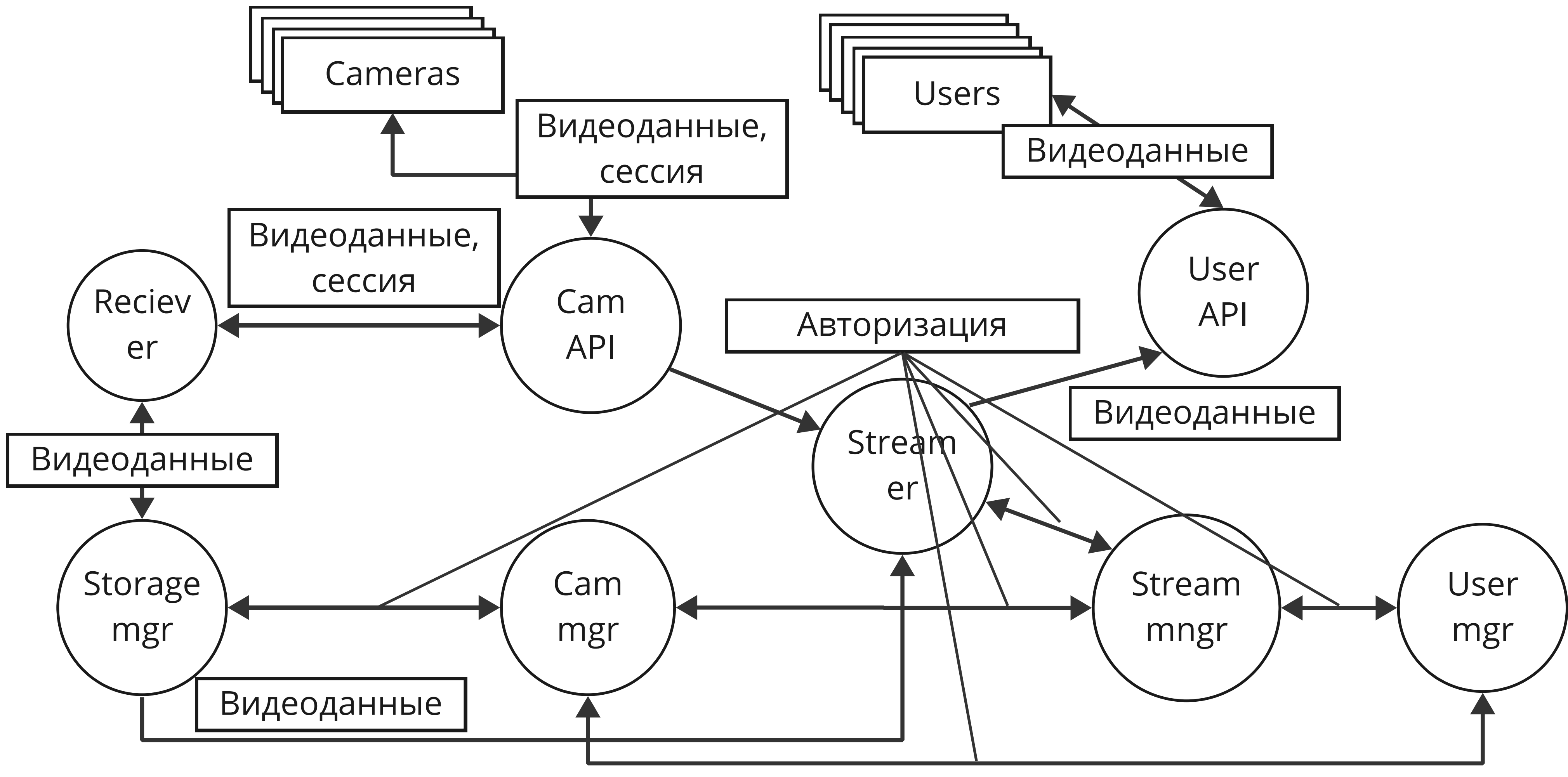


Анализ активов

- Персональные данные, креды, платежные данные – высокий
- Информация с камер наблюдения – средний
- Сообщения с уведомлениями, не содержащими данные пользователя – низкий
- Внешние данные – минимальный

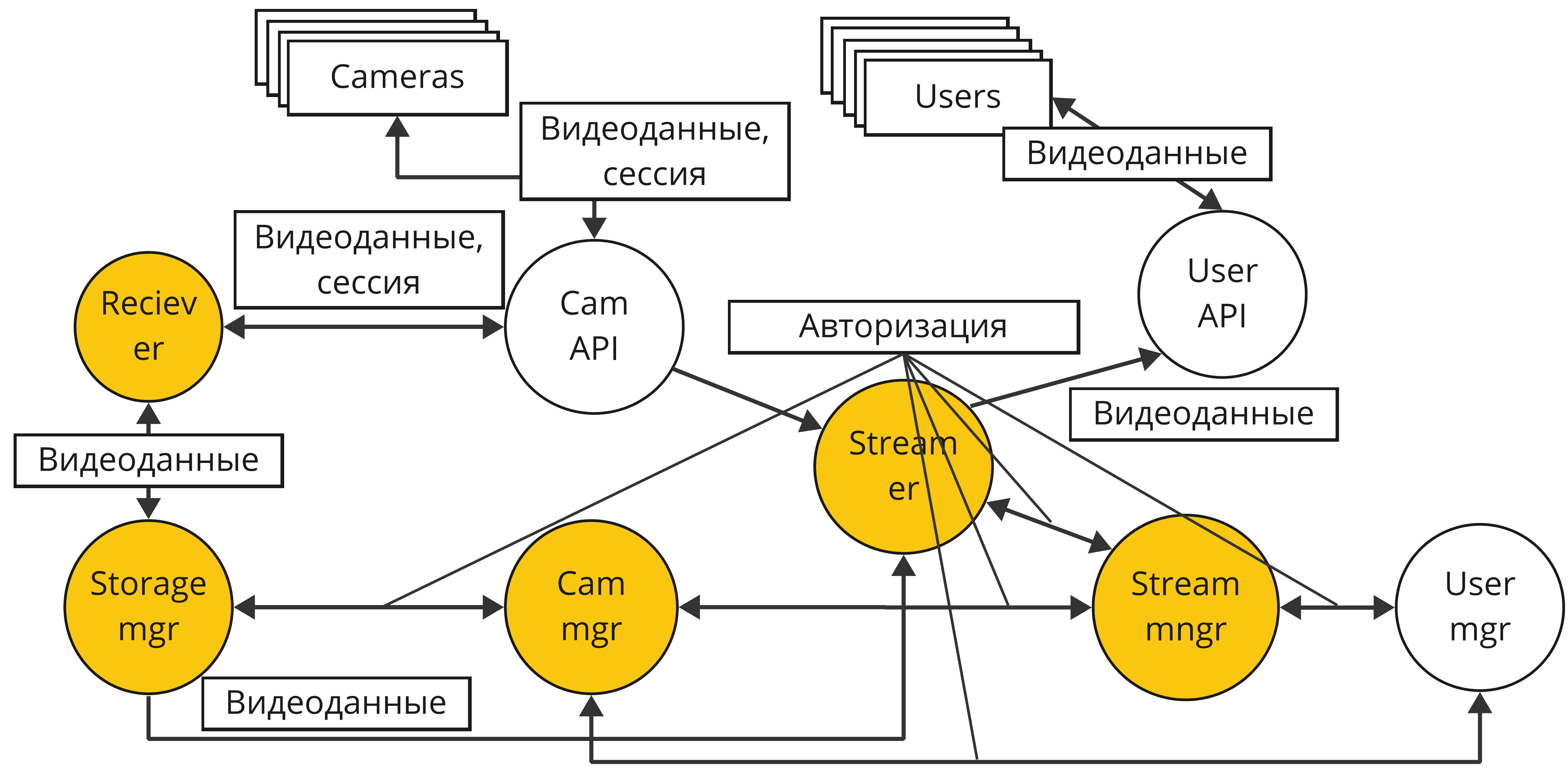


Выделяем активы



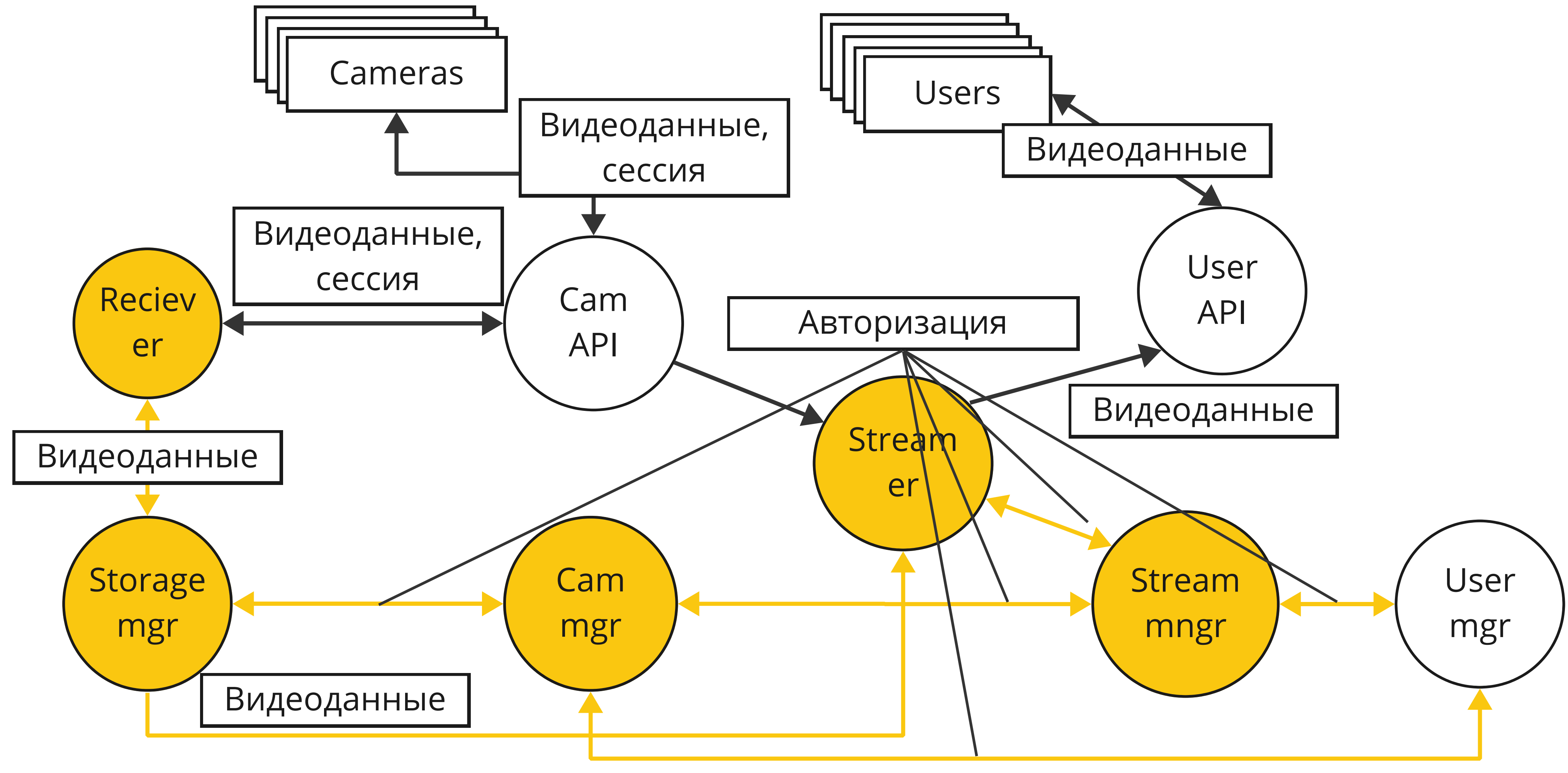
HighLoad++ 2022

Наносим результаты



HL HighLoad++ 2022

Оценка потоков данных



HighLoad++ 2022

Выделяем активы

Users

Notification

Transactions

Bank API

Видеоданные, креденшелы, платежная информация, персональная информация данные камеры

Персональная информация

Транзакция

Account

User API

Платежная информация, персональная информация, креды

Платежные данные

Платежные данные подписка

Транзакция

Bank

User mgr

Подписка

Payments

Авторизация



Наносим результаты



Оцениваем потоки

Users

Notificat
ion

Видеоданные, креденшелы, платежная информация, персональная информация данные камеры

Персональная информация

Transac
tions

Транзакция

Bank
API

User
API

Платежная информация, персональная информация, креды

Account

Платежные данные

Транзакция

Платежные данные подписка

Bank

Авторизация

User
mgr

Подписка

Paymen
ts



Общий вид

Cameras
Users
Notification
Reciever
Cam API
User API
Account
Transactions
Bank API
Streamer
Bank
Storage mgr
Cam mgr
Stream mngr
User mgr
Payments



Применим модель Биба

Cameras

Users

Видеоданные, сессия

Видеоданные

Видеоданные, сессия

Reciev er

Cam API

User API

Авторизация

Видеоданные

Stream er

Видеоданные

Storage mgr

Cam mgr

Stream mngr

User mgr

Видеоданные



Применяем модель ЛаПадула

Cameras

Users

Видеоданные, сессия

Видеоданные

Видеоданные, сессия

Reciever

Cam API

User API

Авторизация

Streamer

Видеоданные

Видеоданные

Storage mgr

Cam mgr

Stream mngr

User mgr

Видеоданные



Накладываем друг на друга

Cameras

Видеоданные, сессия

Users

Видеоданные

Видеоданные, сессия

Cam API

User API

Reciev er

Авторизация

Видеоданные

Stream er

Видеоданные

Storage mgr

Cam mgr

Stream mngr

User mgr

Видеоданные



Применяем на вторую часть

Users

Notification

Видеоданные, креденшелы, платежная информация, персональная информация данные камеры

Персональная информация

Transactions

Bank API

Транзакция

User API

Платежная информация, персональная информация, креды

Account

Платежные данные

Платежные данные подписка

Транзакция

Bank

User mgr

Подписка

Payments

Авторизация



Вторую модель

Users

Notification

Видеоданные, креденшелы, платежная информация, персональная информация данные камеры

Персональная информация

Transactions

Bank API

Транзакция

User API

Платежная информация, персональная информация, креды

Account

Платежные данные

Транзакция

Платежные данные подписка

Bank

Авторизация

User mgr

Подписка

Payments



Накладываем друг на друга

Users

Notification

Видеоданные, креденшелы, платежная информация, персональная информация данные камеры

Персональная информация

Transactions

Bank API

Транзакция

User API

Платежная информация, персональная информация, креды

Account

Платежные данные

Платежные данные подписка

Транзакция

Bank

Авторизация

User mgr

Подписка

Payments



Общий вид

Cameras
Users
Notificat
ion
Reciev
er
Cam
API
User
API
Account
Transac
tions
Bank
API
Stream
er
Bank
Storage
mgr
Cam
mgr
Stream
mngr
User
mgr
Paymen
ts



Утечка в логи

Storage mgr
Account
Notification
Transactions
Payments

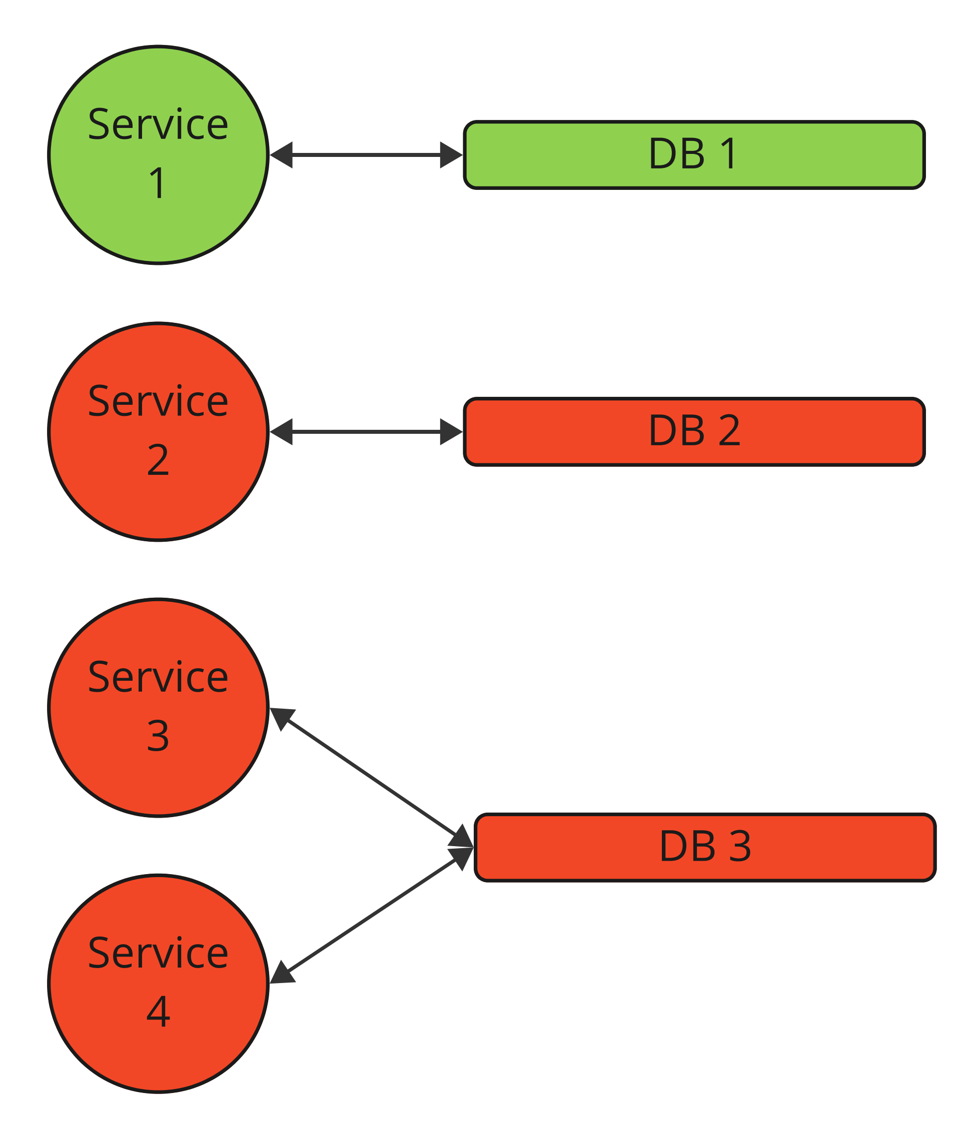
LOG

Reciever
Camera mgr
Streamer
Stream mgr
User mgr



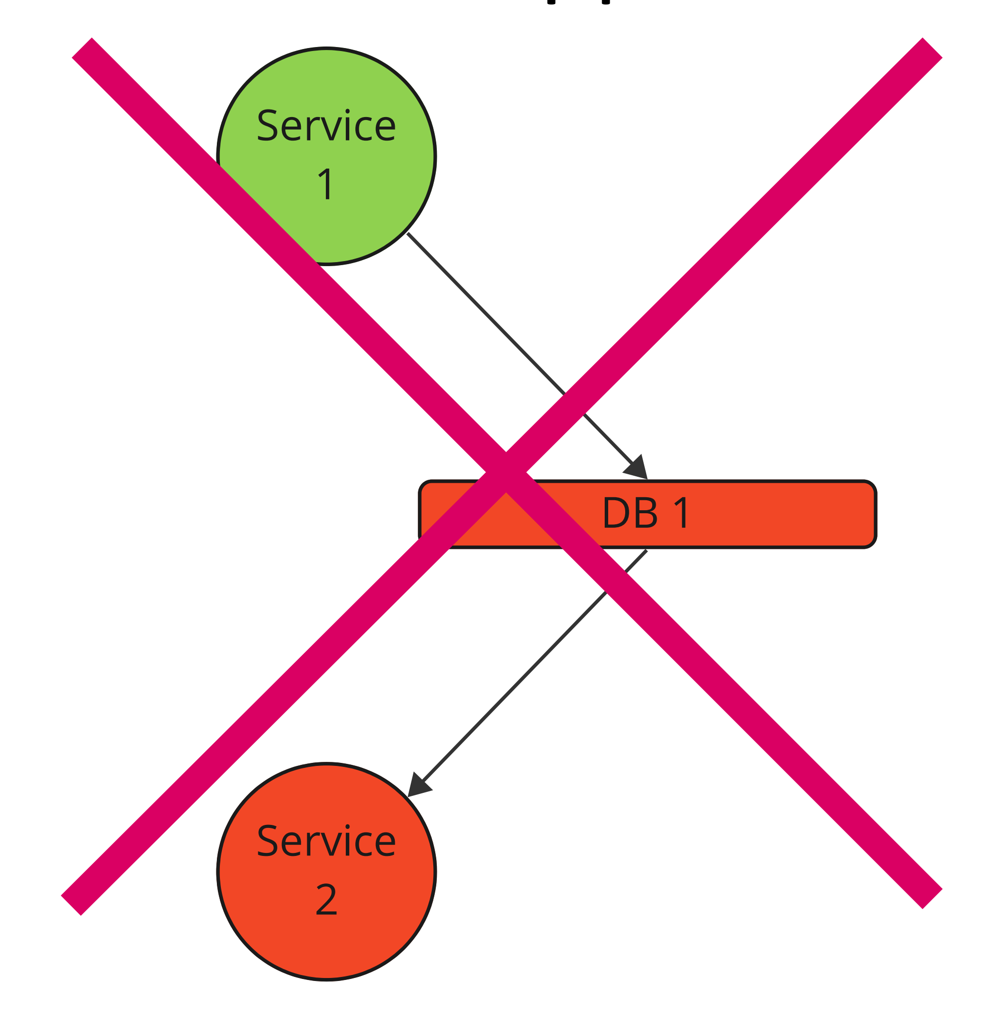
О чем мы забыли?

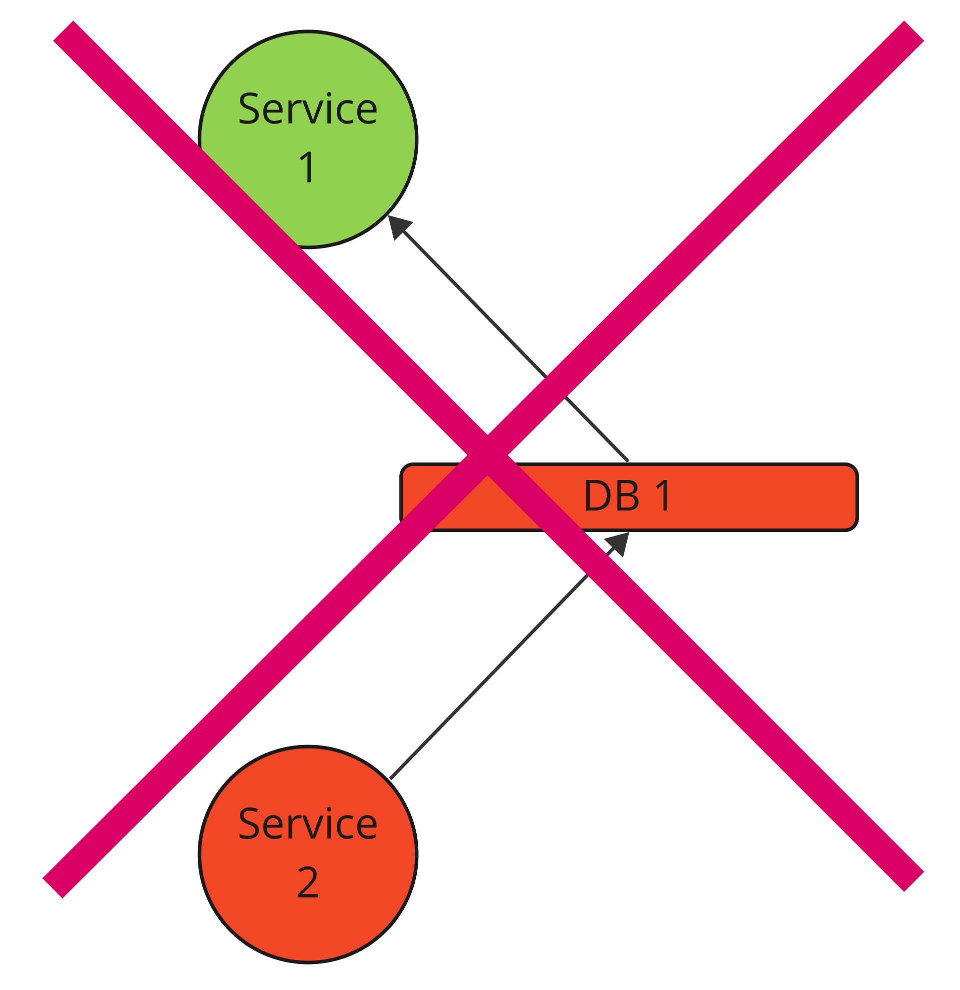
Хранилища данных для сервисов





Как нельзя делать?







Как можно делать?

Service 1

Service 4

DB 1

DB 3

Service 2

DB 2

Service 3



Как правильно делать

Service 1
DB 1
Service 3
Service 2
DB 2
Service 4
DB 3



Простое написание Network policy

```yaml
spec:
  podSelector:
    matchLabels:
      app: Transactions
  ingress:
    - from:
      - podSelector:
        matchLabels:
          app: Payments
  egress:
    - to:
      - podSelector:
        matchLabels:
          app: Payments
```





Как избавиться от звезды смерти?





Плюсы такого подхода

- Низкий порог входа
- Контроль путей приема и передачи информации системой (и неочевидные)
- Границы информации внутри системы
- Существенное затруднение горизонтального продвижения в случае атак



Помогало ли это в жизни?



Спасибо за внимание!



Голосуйте за мой доклад

Давайте обсудим плюсы и минусы подхода

bit.ly/archsecurity

Алексей Федулаев
tg: @int0x80h



